Multiple-Biometric Evaluation (MBE) 2010

Still Face Image Concept, Evaluation Plan and API

Version 0.5

Patrick Grother

Image Group
Information Access Division
Information Technology Laboratory
National Institute of Standards and Technology

November 16, 2009

Status of this Document

The entire content of this document is open for comment. Comments and questions should be submitted to mbe2010@nist.gov.

Intended Timeline of the MBE-STILL Evaluation

July to August 2010	NIST documentation and reports released
January 26 to May 14, 2010	Still-face open submission period
December 17, 2009	Final evaluation plan
December 11, 2009	Comments period closes on second draft of this document.
December 04, 2009	MBGC Workshop, Washington DC
December 03, 2009	Second draft evaluation plan (revised version of this document) for public comment.
November 30, 2009	Request that participants give non-binding no-commitment indication of whether they will participate in the test.
	Comments period closes on first draft of this document.
November 13, 2009	Initial draft evaluation plan (this document) for public comment.

October 2009

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

November 2009

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

December 2009

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

January 2010

Su	Mo	Tu	We	Th	Fr	Sa
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

February 2010

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28						

March 2010

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

April 2010

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

May 2010

Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

June 2010

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

July 2010

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Table of Contents

List of Figures

List of Tables

Acknowledgements

Project History

Terms and definitions

The abbreviations and acronyms of Table 1 are used in many parts of this document.

Table 1 – Abbreviations

FNIR	False negative identification rate
FPIR	False positive identification rate
FMR	False match rate
FNMR	False non-match rate
GFAR	Generalized false accept rate
GFRR	Generalized false reject rate
DET	Detection error tradeoff characteristic: For verification this is a plot of FNMR vs. FMR (sometimes as normal deviates, sometimes on log-scales). For identification this is a plot of FNIR vs. FPIR.
INCITS	InterNational Committee on Information Technology Standards
ISO/IEC 19794	Multipart standard of "Biometric data interchange formats"
I385	INCITS 385:2004 - U.S. precursor to the 19794-5 international standard
ANSI/NIST Type 10	The dominant container for facial images in the law enforcement world.
MBE	NIST's Multiple Biometric Evaluation program

NIST	National Institute of Standards and Technology
PIV	Personal Identity Verification
SC 37	Subcommittee 37 of Joint Technical Committee 1 – developer of biometric standards
SDK	The term Software Development Kit refers to any library software submitted to NIST. This is used synonymously with the terms "implementation" and "implementation under test".

1. MBE

1.1. Overview

This document establishes a concept of operations and an application programming interface (API) for evaluation of face recognition implementations submitted to NIST's Multiple Biometric Evaluation. This document covers only the recognition of two-dimensional still-images. As depicted in Figure 1, the recognition-from-video and face-iris portal tracks of the MBE program are documented elsewhere. See http://face.nist.gov/mbe for all MBE documentation.

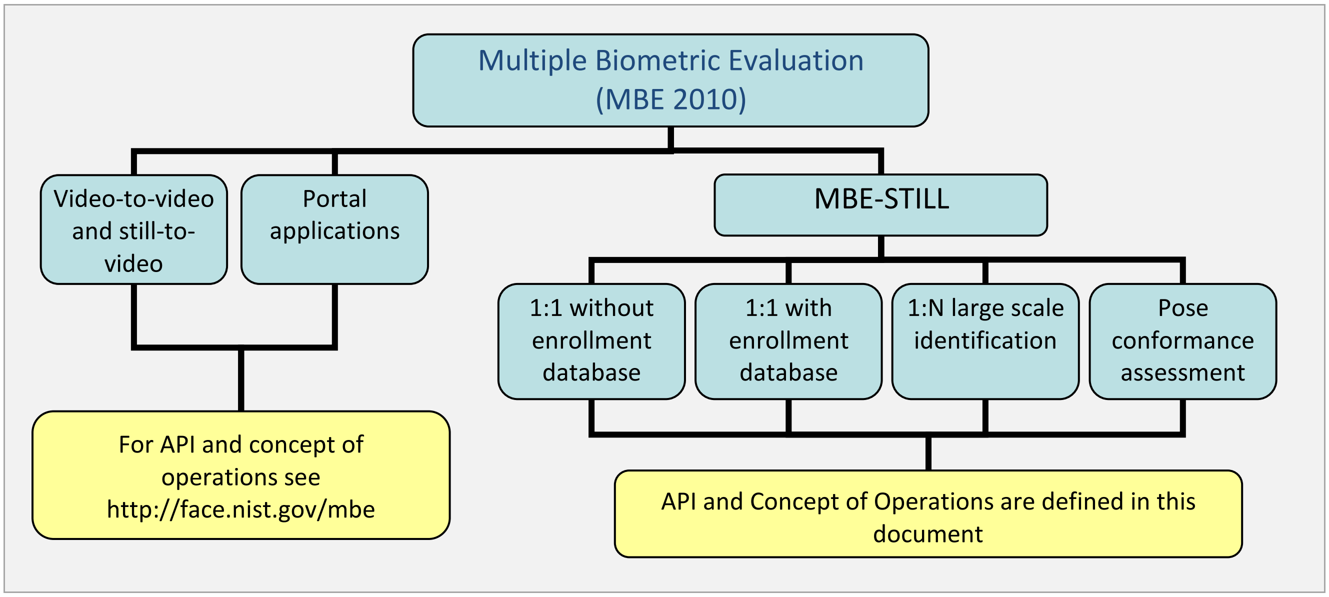
Multiple Biometric Evaluation (MBE 2010)
Video-to-video and still-to-video
Portal applications
MBE-STILL
1:1 without enrollment database
1:1 with enrollment database
1:N large scale identification
Pose conformance assessment
For API and concept of operations see http://face.nist.gov/mbe
API and Concept of Operations are defined in this document

Figure 1- Organization and documentation of the MBE

1.2. Scope

This document is concerned only with 2D still images. The working name for

1.3. Audience

Universities and commercial concerns with capabilities in following areas are invited to participate in the MBE still-face test.

- Identity verification with face recognition algorithms
- Large scale identification implementations.
- Organizations with a capability to assess pose orientation of a face in an image.

Organizations will need to implement the API defined in this document. Participation is open worldwide. There is no charge for participation. While NIST intends to evaluate technologies that could be readily made operational, the test is also open to experimental, prototype and other technologies.

1.4. Market drivers

This test is intended to support a plural marketplace of face recognition systems. While the dominant application, in terms of revenue, has been one-to-many search for driving licenses and visa issuance, the deployment of one-to-one face recognition has re-emerged with the advent of the e-Passport verification projects[1]. In addition, there remains considerable activity in the use of FR for surveillance applications.

[1] These match images acquired from a person crossing a border against the ISO/IEC 19794-5 facial image stored on the embedded ISO/IEC 7816 + ISO/IEC ISO 14443 chips.

These applications are differentiated by the population size (and other variables). In the driving license duplicate detection application, the enrollment database might exceed 10^7 people. In the surveillance application, the watchlist size can readily extend to 10^4.

1.5. Offline testing

While this set of tests is intended as much as possible to mimic operational reality, this remains an offline test executed on databases of images. The intent is to assess the core algorithmic capability of face recognition algorithms. This test will be conducted purely offline - it does not include a live human-presents-to-camera component. Offline testing is attractive because it allows uniform, fair, repeatable, and efficient evaluation of the underlying technologies. Testing of implementations under a fixed API allows for a detailed set of performance related parameters to be measured.

1.6. Phased testing and schedule

To support research and development efforts, this testing activity will embed multiple rounds of testing. Once the test commences, NIST will test implementations on a first-come-first-served basis and will return result to providers as expeditiously as possible. NIST will return results to vendors as soon as they are produced and independently of the other status of other providers' implementations. The results reports will expand as revised implementations are tested.

These test rounds are intended to support improved performance. Each test will result in a "score-card" provided to the participant. The score cards will

- be machine generated (i.e. scripted),
- be provided to participants with identification of their implementation,
- include results from other implementations, but will not identify the other providers.
- be regenerated on-the-fly, primarily whenever any implementation completes testing, or when new analysis is added.

NIST does not intend to release these test reports. NIST may release such information to the U.S. Government test sponsors. While these reports are not intended to be made public, NIST can only request that agencies not release this content.

At some point NIST will terminate the testing rounds and will write a final public report. NIST may publish

- Reports (typically as numbered NIST Interagency Reports)
- Publications in the academic literature
- Presentations (typically PowerPoint).

The final test report will publish results for the best-performing implementation. Because the definition of "best" is ill-defined, the published reports may report results for other implementations. The intention is to report results for the most capable implementations (see section 1.11, on metrics). Other results may be included (e.g. in appendices) to show, for example, examples of progress or tradeoffs.

1.7. Application scenarios

The test will include one-to-one verification tests, and one-to-many identification tests[2]. As described in Table 2, the test is intended to represent:

- Close-to-operational use of face recognition technologies in identification applications which have enrolled populations in excess of one million persons.
- Verification scenarios in which two still images are compared.
- Verification scenarios in which an image is compared with entries in an enrolled database.

[2] NIST has previously only modeled identification scenarios. The simplest simulation mimics a 1:N search by conducting N 1:1 comparisons.

Table 2 – Subtests supported under the MBE still-face activity

#		A	B	C	D
1.	Aspect	1:1 verification	1:1 verification	1:N identification	Pose estimation
2.	Enrollment dataset	None, application to single images.	N enrolled subjects	N enrolled subjects	None, application to single images,
3.	Example application	Verification of e-Passport facial image against a live border-crossing image.	Verification of live capture against a central access control database after presentation of an ID credential	Open-set identification of an image against a central database, e.g. a search of a mugshot against a database of known criminals.	During capture, algorithm assesses whether face is frontal or not, or estimates pose. Frontal pose is required in formal standards because non-frontal pose eventually degrades face recognition accuracy.
4.	Score or feature space normalization support	Vendor applies normalization techniques-are applied against internal implementation-supplied dataset	Vendor applies normalization techniques against enrollment dataset and internal datasets	Any score or feature based statistical normalization techniques-are applied against enrollment database	
5.	Intended number of subjects	Up to O(10^5)	Up to O (10^5)	Up to O(10^7) but O(10^4) will be considered also.	Expected O(10^3)
6.	Number of images per individual	1	1	Variable, see section 1.9.	1
7.	Metadata items	None. Category label?	None. Category label?	Sex, date of image, date of birth, race. These may not be available to the SDK.	

1.8. Options for participation

Prospective participants should read this document including Annex A. Participants must submit an SDK that provides all of the components identified in one or more of the rows of Table 3. All components in a row shall be supplied.

Table 3 – MBE classes of participation

		1	2	3	4
Class	Participation agreement Annex A	1:1 verification without enrollment database	1:1 verification with enrollment database	1:N identification	Pose conformance estimation
A	+	+			
B	+	+	+		
C	+	+		+	
D	+				+
API		3.1 + 3.2 + 3.3	3.1 + 3.2 +3.5	3.1 + 3.2 +3.4	3.1 + 3.2 +3.6

A participant may enter SDKs for more than one class. Entries in each row are entirely separate, so a library submitted to MBE-STILL will not combine the functionalities of separate rows. That is, each SDK will support exactly one row A, B, C or D. A participant shall not enter class A and B.

Class A might be preferred by submissions from academic institutions because it supports the elemental hypothesis testing verification function "are the images from the same person or not?"

1.9. Use of multiple images per person

Some of the proposed datasets includes K > 2 images per person. This affords the possibility to model a recognition scenario in which a new image of a person is compared against all prior images[3]. Use of multiple images per person has been shown to elevate accuracy over a single image [FRVT 2002b].

For this test, NIST will enroll K ≥ 1 images under each identity. NIST will verify or identify a single image against this identity, and this will ordinarily be the most recent image. The method by which the face recognition implementation exploits multiple images is not regulated: The test seeks to evaluate vendor provided technology for multi-instance fusion. This departs from some prior NIST tests in which NIST executed fusion algorithms ([e.g. [FRVT2002b], and sum score fusion, for example, [MINEX]).

This document defines a template to be the result of applying feature extraction to a set of K ≥ 1 images. That is, a template comes from one or more images, not generally just one.

The number of images per person will depend on the application area:

- In civil identity credentialing (e.g. passports, driving licenses) the images will be acquired approximately uniformly over time (e.g. five years for a Canadian passport). While the distribution of dates for such images of a person might be assumed uniform, a number of factors might undermine this assumption[4].
- In criminal applications the number of images would depend on the number of arrests[5]. The distribution of dates for arrest records for a person (i.e. the recidivism distribution) has been modeled using the exponential distribution, but is recognized to be more complicated. NIST currently estimates that the number of images will never exceed 100.

NIST will not use this API for video data.

1.10. Provision of photograph date information to the implementation

Due to face ageing effects, the utility of any particular enrollment image is dependent on the time elapsed between it and the probe image. In MBE, NIST intends to use the most recent image as the probe image, and to use the remaining prior images under a single enrolled identity.

1.11. Core accuracy metrics

Notionally the error rates for verification applications will be false match and false non-match error rates, FMR and FNMR. Under the ISO/IEC 19795-1 biometric testing and reporting standard, the test must account for "failure to acquire" and "failure to enroll" events (e.g. elective refusal to make a template, or fatal errors). The appropriate metrics will then be generalized error rates (GFAR, GFRR). The two will be equivalent if no failures are observed.

For identification testing, the test will target open-universe applications such as benefits-fraud and watch-lists. It will not address the closed-set task because it is operationally uncommon.

While some one-to-many applications operate with purely rank-based metrics, this test will primarily target score-based identification metrics. Metrics are defined in Table 4. The analysis will survey over various rank and thresholds. Plots of the two error rates, parametric on threshold, will be the primary reporting mechanism.

Table 4 - Summary of accuracy metrics

	Application	Metric		
A	1:1 Verification	FMR	=	Fraction of impostor comparisons that produce an similarity score greater than a threshold value

[3] For example, if a banned driver applies for a driving license under a new name, and the local driving license authority maintains a driving license system in which all previous driving license photographs are enrolled, then the fraudulent application might be detected if the new image matched any of the prior images. This example implies one (elemental) method of using the image history.

[4] For example, a person might skip applying for a passport for one cycle (letting it expire). In addition, a person might submit identical images (from the same photography session) to consecutive passport applications at five year intervals.

[5] A number of distributions have been considered to model recidivism, see ``Random parameter stochastic process models of criminal careers.'' In Blumstein, Cohen, Roth & Visher (Eds.), Criminal Careers and Career Criminals, Washington, D.C.: National Academy of Sciences Press, 1986.

		FNMR	=	Fraction of genuine comparisons that produce a similarity score less than some threshold value
B	1:N Identification Primary identification metric.	FPIR	=	Fraction of searches that do not have an enrolled mate for which one or more candidate list entries exceed a threshold
		FNIR	=	Fraction of searches that have an enrolled mate for which the mate is below a threshold
C	1:N Identification (with rank criteria) Secondary identification metric	FPIR	=	Fraction of searches that do not have an enrolled mate for which one or more candidate list entries exceed a threshold
		FNIR	=	Fraction of searches that have an enrolled mate for which the mate is not in the best R ranks *and* at or above a threshold

NOTE: The metric on line B is a special case of the metric on line C: the rank condition is relaxed (R → N). Metric B is the primary metric of interest because the target application does not include a rank criterion.

NIST will extend the analysis in other areas, with other metrics, and in response to the experimental data and results.

1.12. Reporting template size

Because template size is influential on storage requirements and computational efficiency, this API supports measurement of templates size. NIST will report statistics on template size.

1.13. Reporting computational efficiency

As with other tests, NIST will compute and report recognition accuracy. In addition, NIST will also report timing statistics for all core functions of the submitted SDK implementations. This includes feature extraction, and 1:1 and 1:N recognition. For an example of how efficiency can be reported, see the final report of the NIST Iris Exchange test[6].

1.14. Exploring the accuracy-speed trade-space

Organizations may enter two SDKs per class. This is intended to allow an exploration of accuracy vs. speed tradeoffs for face recognition algorithms running on a fixed platform. NIST will report both accuracy and speed of the implementations tested. While NIST cannot force submission of "fast vs. slow" variants, participants may choose to submit variants on some other axis (e.g. "experimental vs. mature") implementations. NIST encourages “fast-less-accurate vs. slow-more-accurate” with a factor of three between the speed of the fast and slow versions.

1.15. Hardware and software

NIST intends to execute the test on high-end PC-class computers. These machines have 4-cpus, each of which has 4 cores. This allows, for example, 16 processes to be run without time slicing. Each machine has 192GB of main random access memory. All submitted implementations shall run on either

- RedHat Linux Enterprise 5 platforms, based on later linux 2.6 kernels, with gcc 4.3, or
- Windows Server 2008 R2 OS (with linking done with gcc 4.3 under the cygwin[7] layer).

The Linux option is preferred by NIST, but providers should choose whichever platform suits them. Providers are cautioned that their choice of operating system (Linux, Windows) may have some impact on efficiency. NIST will provide appropriate caveats to the test report. NIST will respond to prospective participants' questions on the hardware, by amending this section.

NIST is recommending use of 64 bit implementations throughout. This will support large memory allocation - this seems necessary for the 1:N identification task with image counts in the millions. NIST will allow 32 bit operation for 1:1.

If all templates were to be held in memory, the 192GB capacity implies a limit of < 20KB per template, for a 10 million image enrollment. The API allows read access of the disk during the 1:N search.

[6] See NIST Interagency Report 7269 linked from http://iris.nist.gov/irex

[7] According to http://www.cygwin.com/ is a Linux-like environment for Windows. It consists of two parts: A DLL (cygwin1.dll) which acts as a Linux API emulation layer providing substantial Linux API functionality; a collection of tools which provide Linux look and feel.

1.16. Threaded computations

Table 5 shows the allowed use of multi-threading. In many cases threading is "Not permitted" because NIST will parallelize the test by dividing the workload across many cores and many machines. For the 1:N test, we assume that an implementation that does not thread will be uncompetitive with regards to speed.

Table 5 – Requirements on the use of threaded applications

1	2	3	4	5
Function	1:1 verification without enrollment database	1:1 verification with enrollment database	1:N identification	Pose conformance estimation
Feature extraction	Not permitted	Not permitted	Not permitted	Not permitted
Verification	Not permitted	Not permitted	NA	
Identification	NA	NA	Strongly advised	

1.17. Time limits

The elemental functions of the implementations shall execute under the time constraints of Table 6. Assuming the times are random variables, no hard limits are imposed, so these limits are 90-th percentiles.

Table 6 – Time limits in milliseconds on a single PC core

1	2	3	4	5
Function	1:1 verification without enrollment database	1:1 verification with enrollment database	1:N identification	Pose conformance estimation
Feature extraction enrollment	800	800	500	500
Feature extraction verification	800	800	800	
Verification	5		NA	
Identification per 1,000,000 records on 16 cores	NA	NA	5000	

Editor's NOTE: Are these numbers achievable? Would they be better stated another way?

1.18. Test datasets

This section under is under development. The data has, in some cases, been estimated from initial small partitions. The completion of this section depends on further work. The information is subject to change. We intend to update this section as fully as possible.

NIST is likely to use other datasets in addition.

Table 7 – Main image corpora (others will be used)

	Sandia	FRVT 2002+2006 / HCINT	Multiple Encounter Database
Collection, environment	See the FRGC FAQs for details on these images. See also FRVT 2006 Report, Phillips et al. NIST IR 7408.	Visa application process	Law enforcement booking
Live scan, Paper		Live	Live, few paper
Documentation		See NIST IR 6965 [FRVT2002]	See NIST Special Database 31 Volume 1, available 11/09.
Compression		JPEG mean size 9467 bytes. See [FRVT2002b]	JPEG ~ 20:1
Maximum image size		300 x 252	Mixed, some are 640x480 others are 768x960
Minimum image size		300 x 252	
Eye to eye distance		Median = 71 pixels	mean=156, sd=46
Frontal		Yes, well controlled	Moderately well controlled Profile images will be included and labeled as such.

Full frontal geometry		Yes, in most cases. Faces may have small background than ISO FF requires.	Mostly not. Varying amounts of the torso are visible.
Intended use	1:1	1:1 and 1:N	1:N

1.19. Ground truth integrity

Some of the test databases will be derived from operational systems. They may contain ground truth errors in which

- a single person is present under two different identifiers, or
- two persons are present under one identifier, or
- in which a face is not present in the image.

If these errors are detected, they will be removed. NIST will use aberrant scores (high impostor scores, low genuine scores) to detect such errors. This process will be imperfect, and residual errors are likely. For comparative testing, identical datasets will be used and the presence of errors has is not inherently unfair. For prediction of operational performance, the presence of errors gives incorrect estimates of performance.

2. Data structures supporting the API

2.1. Overview

This section describes separate APIs for the core face recognition applications described in section 1.7. All SDK's submitted to MBE shall implement the functions below here as required by the classes of participation listed in Table 3.

2.2. Requirement

MBE participants shall submit an SDK which implements the "C" prototyped interface of clause 3.

2.3. File formats and data structures

2.3.1. Overview

In this face recognition test, an individual is represented by $K \geq 1$ two-dimensional facial images, and by subject and image-specific metadata.

2.3.2. Dictionary of terms describing images

Images will be accompanied by one of the labels given in Table 8.

Table 8 – Labels describing types of images

	Label as "C" char * string	Meaning
1.	"unknown"	Either the label is unknown or unassigned.
2.	"sandia"	
3.	"visa"	Either a member of the FRVT 2002/2006 HCINT corpus or one of similar properties.
4.	"mugshot"	Either a member of the Multi-encounter law enforcement database or one of similar properties. The image is nominally frontal.
5.	"profile"	The image is a profile image taken from the multi-encounter law enforcement database.

2.3.3. Data structures for encapsulating multiple images

The standardized formats for facial images are the ISO/IEC 19794-5:2005 and the ANSI/NIST ITL 1-2007 type 10 record. The ISO record can store multiple images of an individual in a standalone binary file. In the ANSI/NIST realm, K images of an individual are usually represented as the concatenation of one Type 1 record + K Type 10 records. The result is usually stored as an EFT file.

For the current test, neither ANSI/NIST Type 10 nor ISO/IEC 19794-5 is used because they do not encode metadata information such as capture date and sex[8].

An alternative method of representing K images of an individual is to define a structure containing an image filename and metadata fields. Each file contains a standardized image format, e.g. PNG (lossless) or JPEG (lossy).

Table 9 – Structure for a single face, with metadata

	"C" code fragment	Remarks	
1.	typedef struct sface		
2.	{		
3.	uint8_t *pngdata;	Pointer to a PNG file read into memory	Only one of these will not be NULL.
4.	uint8_t *jpgdata;	Pointer to a JPG file read into memory	
5.	char *filename;	File containing a single image of a face	
6.	char *description;	Single description of the image. The allowed values for this string are given in Table 8.	
7.			
8.	uint16_t mob;	Month of birth (e.g. 1-12); 0 indicates unknown	
9.	uint16_t yob;	Year of birth (e.g. 1964); 0 indicates unknown	
10.	uint16_t month;	Month of image capture [1-12]; 0 indicates unknown	
11.	uint16_t year;	Year of image capture (e.g. 2002); 0 indicates unknown	
12.	uint8_t sex;	This field uses the ISO/IEC 19794-5 values: Unspecified = 0x00; Male = 0x01; Female = 0x02; Unknown 0xFF	
13.	uint8_t race;	This field will use these values: 0x00 - Unassigned or unknown 0x01 -- American Indian or Alaska Native 0x02 -- Asian 0x03 -- Black or African American 0x04 -- Hispanic or Latino 0x05 -- Native Hawaiian or Other Pacific Islander 0x06 -- White	
14.	uint16_t weight;	Body weight in kilograms: 0x00 - unassigned or unknown	
15.	uint16_t height;	Height in meters: 0x00 - unassigned or unknown	
16.	} ONEFACE;		

Table 10 – Structure for a set of images from a single person

	"C" code fragment	Remarks
1.	typedef struct mface	
2.	{	
3.	unsigned int numfaces;	The number of accessible files, F, such that the last element is faces[F-1]
4.	ONEFACE **faces;	Pointers to F pre-allocated face images of the same person.
5.	} MULTIFACE;	

2.3.4. Data structure for eye coordinates

SDKs should return eye coordinates of each enrolled facial image. This function, while not necessary for a recognition test, will assist NIST in assuring the correctness of the test database. The primary mode of use will be for NIST to inspect images for which eye coordinates are not returned, or differ between vendor SDKs.

The eye coordinates shall follow the placement semantics of the ISO/IEC 19794-5:2005 standard - the geometric midpoints of the endocanthion and exocanthion (see clause 5.6.4 of the ISO standard).

Table 11 – Structure for a pair of eye coordinates

	"C" code fragment	Remarks

[8] In ANSI/NIST such content is routinely transmitted in Type 2, but it's not uniformly standardized, and in case overly complicated for the purpose of testing.

1.	`typedef struct ohos`	
2.	`{`	
	`uint8_t  failed;`	If the eye coordinates have been computed and assigned, this value should be set to 0 otherwise it should be set on [1,255].
3.	`int16_t  xleft;`	X and Y coordinate of the center of the subject's left eye. Out-of-range values (e.g. x < 0 or x >= width) indicate the implementation believes the eye center is outside the image.
4.	`int16_t  yleft;`	
5.	`int16_t  xright;`	X and Y coordinate of the center of the subject's right eye. Out-of-range values (e.g. x < 0 or x >= width) indicate the implementation believes the eye center is outside the image.
6.	`int16_t  yright;`	
7.	`} EYEPAIR;`	

2.3.5. Data type for similarity scores

Identification and verification functions shall return a measure of the similarity between the face data contained in the two templates. The datatype shall be an eight byte double precision real. The legal range is [0, DBL_MAX], where the DBL_MAX constant is larger than practically needed and defined in the <limits.h> include file. Larger values indicate more likelihood that the two samples are from the same person.

Providers are cautioned that algorithms that natively produce few unique values (e.g. integers on [0,127]) will be disadvantaged by the inability to set a threshold precisely, as might be required to attain a false match rate of exactly 0.0001, for example.

2.3.6. Data structure for result of an identification search

All identification searches shall return a candidate list of length 50. The list shall be sorted with the most similar matching entries list first with lowest rank. The data structure shall be that of Table 12.

Table 12 – Structure for a candidate list

	"C" code fragment	Remarks
1.	`typedef struct candidate`	
2.	`{`	
3.	`uint8_t failed;`	If the candidate computation failed, this value is set on [1,255]. If the candidate is valid it should be set to 0.
4.	`uint32_t enrollment_seq_id;`	Position in linear list of templates provided to the finalize_enrollment function in Table 22.
5.	`double similarity_score;`	Measure of similarity between the identification template and the enrolled candidate. Higher scores mean more likelihood that the samples are of the same person. An algorithm is free to assign any value to a candidate. The distribution of values will have an impact on the appearance of a plot of false-negative and false-positive identification rates.
6.	`double probability;`	An estimate of the actual probability that the biometric data and candidate belong to *different* persons. This value shall be on [0:1]. Editor's NOTE: Should this be present? Should it be optional?
7.	`uint8_t match;`	Decision - this boolean value gives the implementation's best guess at whether this candidate is a match. The allowed values are as follows: 0x00 → "not a match"; 0x01 → "a match"
8.	`} CANDIDATE;`	

3. API Specification

3.1. Implementation identifiers

All implementations shall support the self-identification function of Table 13. This function is required to support internal NIST book-keeping. The version numbers should be distinct between any versions which offer different algorithmic functionality.

Table 13 – Implementation identifiers

Prototype	int32_t get_pid(	
	uint32_t *nist_assigned_identifier,	Output
	char *sdk_support,	Output
	char *email_address);	Output
Description	This function retrieves an identifier that the provider must request from NIST irex@nist.gov, and hardwire into the source code. NIST will assign the identifier that will uniquely identify the supplier and the SDK version number.	
Output Parameters	nist_assigned_identifier	A PID which identifies the SDK under test. The memory for the identifier is allocated by NIST's calling application, and shall not be allocated by the SDK. The value of the PID will be assigned by NIST to participants. PIDs are available by request to mbe2010@nist.gov.
	sdk_support	The SDK shall implement one of the elemental application scenarios. The SDK will return one of the following values as a null-terminated string. "1:1_PURE" or "1:1_ENROL_DB" or "1:N" or "POSE_ESTIM" NIST will pre-allocate sufficient space.
	email_address	Point of contact email address as null terminated ASCII string. NIST will allocate at least 64 bytes for this. SDK shall not allocate.
Return Value	0	Success
	Other	Vendor-defined failure

3.2. Maximum template size

All implementations shall report the maximum expected template sizes. These values will be used by the NIST test harnesses to pre-allocate template data. The values should apply to a single image. For a K byte MULTIFACE, NIST will allocate K times the value returned. The function call is given in Table 14.

Table 14 - Implementation template size requirements

Prototype	int32_t get_max_template_sizes(	
	uint32_t *max_enrollment_template_size,	Output
	uint32_t *max_recognition_template_size)	Output
Description	This function retrieves the maximum template size needed by the feature extraction routines.	
Output Parameters	max_enrollment_template_size	The maximum possible size of the memory needed to store feature data from a single enrollment image.
	max_recognition_template_size	The maximum possible size of the memory needed to store feature data from a single verification or identification image.
Return Value	0	Success
	Other	Vendor-defined failure

3.3. 1:1 Verification without enrollment database

3.3.1. Overview

The 1:1 testing will proceed in three phases: preparation of enrolment templates; preparation of verification templates; and matching. These are detailed in Table 15.

Table 15 – Functional summary of the 1:1 application

Phase	#	Name	Description	Performance Metrics to be reported by NIST
Initialization	I1	Initialization	Function to allow implementation to read configuration data, if any.	None
Enrollment	E1	Serial enrolment	Given K ≥ 1 input images of an individual, the implementation will create a proprietary enrollment template. NIST will manage storage of these templates. NIST requires that these operations may be executed in a loop in a single process invocation, or as a sequence of independent process invocations, or a mixture of both.	Statistics of the time needed to produce a template. Statistics of template size. Rate of failure to produce a template and rate of erroneous function.
Verification	V1	Serial verification	Given K ≥ 1 input images of an individual, the implementation will create a proprietary verification template. NIST will manage storage of these templates. NIST requires that these operations may be executed in a loop in a single process invocation, or as a sequence of independent process invocations, or a mixture of both.	Statistics of the time needed to produce a template. Statistics of template size. Rate of failure to produce a template and rate of erroneous function.
Matching (i.e. comparison)	C1	Serial matching	Given one proprietary enrollment template and one proprietary verification template, compare these and produce a similarity score. NIST requires that these operations may be executed in a loop in a single process invocation, or as a sequence of independent process invocations, or a mixture of both.	Statistics of the time taken to compare two templates. Accuracy measures, primarily reported as DETs.

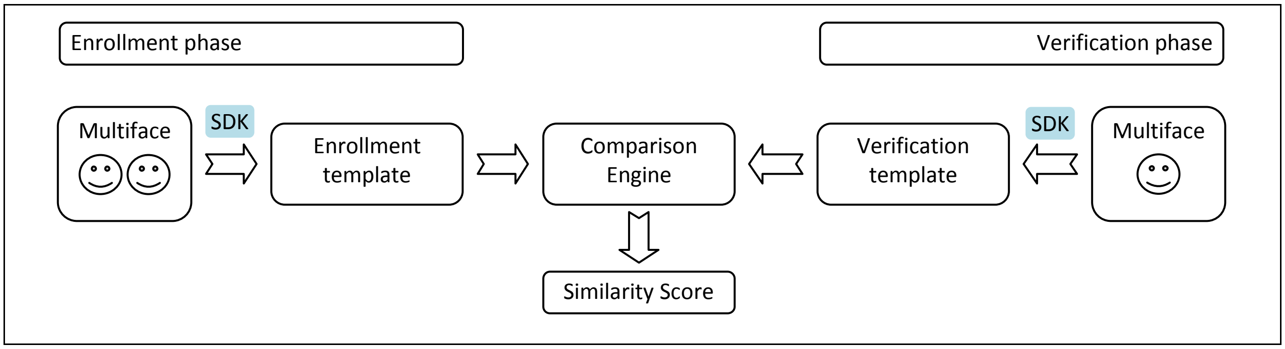

Figure 2- Schematic of verification without enrollment database

3.3.2. API

3.3.2.1. Initialization of the implementation

Before any template generation or matching calls are made, the NIST test harness will make a call to the initialization of the function in Table 16.

Table 16 – SDK initialization

Prototype	int32_t initialize_verification(	
	const char *configuration_location	Input
	const char **descriptions,	Input
	const uint8_t num_descriptions);	Input
Description	This function initializes the SDK under test. It will be called by the NIST application before any call to convert_multiface_to_enrollment_template. The SDK under test should set all parameters. The SDK should	
Input Parameters	configuration_location	A read-only directory containing any vendor-supplied configuration parameters or run-time data files. The name of this directory is assigned by NIST. It is not hardwired by

		the provider. The names of the files in this directory are hardwired in the SDK and are unrestricted.
	descriptions	A lexicon of labels one of which will be assigned to each enrollment image. EXAMPLE: The descriptions could be {"mugshot", "visa"}.
	num_descriptions	The number of items in the description. In the example above this is 2.
Output Parameters	none	
Return Value	0	Success
	2	Vendor provided configuration files are not readable in the indicated location.
	Other	Vendor-defined failure

3.3.2.2. Template generation

The functions of Table 17 support role-specific generation of a template data. The format of the templates is entirely proprietary.

Table 17 – Template generation

Prototypes	int32_t convert_image_to_enrollment_template(	
	const MULTIFACE *input_faces,	Input
	uint32_t *template_size,	Output
	uint8_t *proprietary_template);	Output
	int32_t convert_image_to_verification_template(	
	const MULTIFACE *input_faces,	Input
	uint32_t *template_size,	Output
	uint8_t *proprietary_template);	Output
Description	This function takes a MULTIFACE, and outputs a proprietary template. The memory for the output template is allocated by the NIST test harness before the call i.e. the implementation shall not allocate memory for the result. In all cases, even when unable to extract features, the output shall be a template record that may be passed to the match_templates function without error. That is, this routine must internally encode "template creation failed" and the matcher must transparently handle this.	
Input Parameters	input_faces	An instance of a Table 10 structure. Implementations must alter their behavior according to the number of images contained in the structure.
Output Parameters	template_size	The size, in bytes, of the output template
	proprietary_template	The output template. The format is entirely unregulated. NIST will allocate a KT byte buffer for this template: The value K is the number of images in the MULTIFACE; the value T is output by the maximum template size functions of Table 14.
Return Value	0	Success
	2	Elective refusal to process this kind of MULTIFACE
	4	Involuntary failure to extract features (e.g. could not find face in the input-image)
	6	Elective refusal to produce a template (e.g. insufficient pixels between the eyes)
	8	Cannot parse input data (i.e. assertion that input record is non-conformant)
	Other	Vendor-defined failure. Failure codes must be documented and communicated to NIST with the submission of the implementation under test.

3.3.2.3. Matching

Matching of one enrollment against one verification template shall be implemented by the function of Table 18.

Table 18 – Template matching

Prototype	int32_t match_templates(	
	const uint8_t *verification_template,	Input
	const uint32_t verification_template_size,	Input
	const uint8_t *enrollment_template,	Input
	const uint32_t enrollment_template_size,	Input

	double *similarity);	Output
Description	This function compares two opaque proprietary templates and outputs a non-negative match score. The returned score is a non-negative distance measure. It need not satisfy the metric properties. NIST will allocate memory for this parameter before the call. When either or both of the input templates are the result of a failed template generation (see Table 10), the dissimilarity score shall be -1 and the function return value shall be 2.	
Input Parameters	verification_template	A template from create_template().
	verification_template_size	The size, in bytes, of the input verification template $0 \le N \le 2^{16} - 1$
	enrollment_template	A template from create_template().
	enrollment_template_size	The size, in bytes, of the input enrollment template $0 \le N \le 2^{16} - 1$
Output Parameters	similarity	A similarity score resulting from comparison of the templates, on the range [0,DBL_MAX]. See section 2.3.5.
Return Value	0	Success
	2	Either or both of the input templates were result of failed feature extraction
	Other	Vendor-defined failure

3.4. 1:N Identification

3.4.1. Overview

The 1:N application proceeds in two phases, enrollment and identification. The identification phase includes separate pre-search feature extraction stage, and a search stage.

The design reflects the following objectives for testing 1:N implementations.

- support distributed enrollment on multiple machines, with multiple processes running in parallel
- allow recovery after a fatal exception, and measure the number of occurrences
- ability to re-locate enrollment data onto many machines to support parallel testing
- respect the black-box nature of biometric templates
- extend complete freedom to the provider to use arbitrary algorithms
- support measurement of duration of core function calls
- support measurement of template size

Table 19 – Structure for the proposed formats for ISO/IEC 19794-6

Phase	#	Name	Description	Performance Metrics to be reported by NIST
Enrollment	E1	Initialization	Give the implementation advance notice of the number of individuals and images that will be enrolled. Give the implementation the name of a directory where any provider-supplied configuration data will have been placed by NIST. This location will otherwise be empty. The implementation is permitted read-write-delete access to this directory during this phase. After enrollment, NIST may rename and relocate the enrollment directory - the implementation should not depend on the name of the enrollment directory.	

	E2	Parallel Enrollment	For each of N individuals, pass multiple images of the individual to the implementation for conversion to a combined template. The implementation will return a template to the calling application. The implementation is permitted read-only access to the enorllment directory during this phase. NIST's calling application will be responsible for storing all templates as binary files. These will not be available to the implementation during this enrollment phase. Multiple instances of the calling application may run simultaneously or sequentially. These may be executing on different computers. The same person will not be enrolled twice.	Statistics of the times needed to enroll an individual. Statistics of the sizes of created templates. The incidence of failed template creations.
	E3	Finalization	Permanently finalize the enrollment directory. This supports, for example, adaptation of the image-processing functions, adaptation of the representation, writing of a manifest, indexing, and computation of statistical information over the enrollment dataset. The implementation is permitted read-write-delete access to the enrollment directory during this phase.	Size of the enrollment database as a function of population size N and the number of images.
Pre-search	S1	Template preparation	For each probe, create a template from a set of input images. This operation will generally be conducted in a separate process invocation to step S2. The implementation is permitted no access to the enrollment directory during this phase. The result of this step is a search template.	Statistics of the time needed for this operation. Statistics of the size of the search template.
Search	S2	Initialization	Give the implementation a location of an enrollment directory. The implementation should read all or some of the enrolled data into main memory, so that searches can commence. The implementation is permitted read-only access to the enrollment directory during this phase.	
	S3	Search	A template is searched against the enrolment database. The implementation is permitted read-only access to the enrollment directory during this phase.	

3.4.2. Initialization of the enrollment session

Before any enrollment (feature extraction) calls are made, the NIST test harness will make a call to the initialization of the function in Table 20.

Table 20 – Enrollment initialization

Prototype	int32_t initialize_enrollment_session(	
	const char *configuration_location	Input
	const char *enrollment_directory,	Input
	const uint32_t num_persons,	Input
	const uint16_t num_images,	Input
	const char **descriptions,	Input
	const uint8_t num_descriptions);	Input
Description	This function initializes the SDK under test. It will be called by the NIST application before any call to convert_multiface_to_enrollment_template. The SDK under test should set all parameters. The SDK should	
Input Parameters	configuration_location	A read-only directory containing any vendor-supplied configuration parameters or run-time data files.
	enrollment_directory	The directory will be initially empty, but may have been initialized and populated by separate invocations of the enrollment process. When this function is called, the SDK may populate this folder in any manner it sees fit. Permissions will be read-write-delete.

	num_persons	The number of persons who will be enrolled 0 ≤ N ≤ 2^{16} - 1
	num_images	The number of images that will be enrolled.
	descriptions	A lexicon of labels one of which will be assigned to each enrollment image. EXAMPLE: The descriptions could be {"mugshot", "visa"}.
	num_descriptions	The number of items in the description. In the example above this is 2.
Output Parameters	none	
Return Value	0	Success
	2	Either or both of the input templates were result of failed feature extraction
	Other	Vendor-defined failure

3.4.3. Enrollment

A MULTIFACE is converted to an atomic enrollment template using the function of Table 21.

Table 21 – Enrollment feature extraction

Prototypes	int32_t convert_multiface_to_enrollment_template(	
	const MULTIFACE *input_faces,	Input
	EYEPAIR **output_eyes,	Output
	uint32_t *template_size,	Output
	uint8_t *proprietary_template);	Output
Description	This function takes a MULTIFACE, and outputs a proprietary template. The memory for the output template is allocated by the NIST test harness before the call i.e. the implementation shall not allocate memory for the result. If the function executes correctly (i.e. returns a zero exit status), the calling application will store the template to a uniquely named file. This will be done via binary fopen, fwrite, fclose. The file will reside entirely within the enrollment directory. The filenames will be concatenated into a manifest and passed to the enrollment finalization function (see section 3.4.4). The NIST application will create a hierarchy of directories within the enrollment directory. This will be done to support efficient read / write performance given the properties of the filesystem with respect to deep / broad directory trees and number of files per directory. IMPORTANT. The implementation must not attempt writes to the enrollment directory (nor other resources). Any data needed during subsequent searches should be included in the template, or created from the templates during the enrollment finalization function of section 3.4.4.	
Input Parameters	input_faces	An instance of a Table 10 structure. Implementations must alter their behavior according to the number of images contained in the structure.
Output Parameters	output_eyes	For each input image in the MULTIFACE the function shall return the estimated eye centers. The calling application will pre-allocate the correct number of EYEPAIR structures (i.e. one for each image in the MULTIFACE).
	template_size	The size, in bytes, of the output template
	proprietary_template	The format is entirely unregulated. NIST will allocate a KT byte buffer for this template: The value K is the number of images in the MULTIFACE; the value T is output by the maximum enrollment template size function of Table 14.
Return Value	0	Success
	2	Elective refusal to process this kind of MULTIFACE
	4	Involuntary failure to extract features (e.g. could not find face in the input-image)
	6	Elective refusal to produce a template (e.g. insufficient pixels between the eyes)
	8	Cannot parse input data (i.e. assertion that input record is non-conformant)
	Other	Vendor-defined failure. Failure codes must be documented and communicated to NIST with the submission of the implementation under test.

3.4.4. Finalize enrollment

After all templates have been created, the function of Table 22 will be called. This freezes the enrollment data. After this call the enrollment dataset will be forever read-only. This API does not support interleaved enrollment and search phases.

The function allows the implementation to conduct, for example, statistical processing of the feature data, indexing and data re-organization. The function may alter the file structure. It may increase or decrease the size of the stored data. No output is expected from this function, except a return code.

Table 22 – Enrollment finalization

Prototypes	int32_t finalize_enrollment (	
	const char *enrolment directory	Input
	const char **template_names,	Input
	const uint32_t num_template_names);	Input
Description	This function takes the name of the top-level directory where enrollment data was stored. The directory permissions will be read + write. The function also takes an array of filenames of the templates, and the number thereof. The filenames and paths of the templates were assigned by NIST during enrollment. The function supports post-enrollment vendor-optional book-keeping operations and statistical processing. This function should be tolerant of being called two or more times. Second and third invocations should probably do nothing. The function will generally be called in a separate process as the enrollment.	
Input Parameters	enrollment_directory	The top-level directory in which enrollment data was placed. This variable allows an implementation to locate any private initialization data it elected to place in the directory.
	template_names	An array of (char *) filenames. These are relative paths to files. The files will have read-write-delete permission. Each file contains a template, as produced by the "convert_multiface_to_enrollment_template" function of section 3.4.3. The format of all pathnames will be canonical Unix style pathnames using forward slash directory separators.
	num_template_names	The number of template names.
Output Parameters	None	
Return Value	0	Success
	2	Elective refusal to process this kind of MULTIFACE
	4	Involuntary failure to extract features (e.g. could not find face in the input-image)
	6	Elective refusal to produce a template (e.g. insufficient pixels between the eyes)
	8	Cannot parse input data (i.e. assertion that input record is non-conformant)
	Other	Vendor-defined failure. Failure codes must be documented and communicated to NIST with the submission of the implementation under test.

3.4.5. Pre-search feature extraction

A MULTIFACE is converted to an atomic identification template using the function of Table 23. The result may be stored by NIST, or used immediately. The SDK shall not attempt to store any data.

Table 23 – Identification feature extraction

Prototypes	int32_t convert_multiface_to_identification_template(	
	const char *configuration_location,	Input
	const MULTIFACE *input_faces,	Input
	EYEPAIR **output_eyes,	Output
	uint32_t *template_size,	Output
	uint8_t *identification_template);	Output
Description	This function takes a MULTIFACE, and outputs a proprietary template. The function also takes the name of the directory where the vendor-supplied SDK configuration data is stored. The memory for the output template is allocated by the NIST test harness before the call i.e. the implementation shall not allocate memory for the result.	

	If the function executes correctly it returns a zero exit status. The NIST calling application may commit the template to permanent storage, or may keep it only in memory (the vendor implementation does not need to know). The function shall not have access to the enrollment data, nor shall it attempt access.	
Input Parameters	configuration_location	A read-only directory containing any vendor-supplied configuration parameters or run-time data files.
	input_faces	An instance of a Table 10 structure. Implementations must alter their behavior according to the number of images contained in the structure.
Output Parameters	output_eyes	For each input image in the MULTIFACE the function shall return the estimated eye centers. The calling application will pre-allocate the correct number of EYEPAIR structures (i.e. one for each image in the MULTIFACE).
	template_size	The size, in bytes, of the output template
	identification_template	The output template for a subsequent identification search. The format is entirely unregulated. NIST will allocate a buffer of size "max_recognition_template_size" as returned by the function of Table 14.
Return Value	0	Success
	2	Elective refusal to process this kind of MULTIFACE
	4	Involuntary failure to extract features (e.g. could not find face in the input-image)
	6	Elective refusal to produce a template (e.g. insufficient pixels between the eyes)
	8	Cannot parse input data (i.e. assertion that input record is non-conformant)
	Other	Vendor-defined failure. Failure codes must be documented and communicated to NIST with the submission of the implementation under test.

3.4.6. Initialization

The function of Table 24 will be called once prior to one or more calls to the searching function of Table 25.

Table 24 - Identification initialization

Prototype	int32_t initialize_identification_session(	
	const char *enrollment_directory);	Input
Description	This function reads whatever content is present in the enrollment_directory, for example a manifest placed there by the finalize_enrollment function.	
Input Parameters	enrollment_directory	The top-level directory in which enrollment data was placed.
Return Value	0	Success
	Other	Vendor-defined failure

3.4.7. Search

The function of Table 25 compares a proprietary identification template against the enrollment data and returns a candidate list.

Table 25 – Identification search

Prototype	int32_t identify_template(	
	const uint8_t *identification_template,	Input
	const uint32_t identification_template_size,	Input
	CANDIDATE *candidate_list[50]);	Output
Description	This function searches a template against the enrollment set, and outputs a list of candidates. The returned candidate_list is a non-negative distance measure. It need not satisfy the metric properties. NIST will allocate memory for this parameter before the call. When either or both of the input templates are the result of a failed template generation, the similarity score shall be -1 and the function return value shall be 2.	
Input Parameters	identification_template	A template from convert_multiface_to_identification_template().
	identification_template_size	The size, in bytes, of the input verification template $0 \le N \le 2^{16} - 1$
Output Parameters	candidate_list	An array of 50 pointers to candidates. The datatype is defined in section 2.3.6.

Return Value	0	Success
	2	Either or both of the input templates were result of failed feature extraction
	Other	Vendor-defined failure

3.5. 1:1 Verification with enrollment database

For verification with an enrollment database, the sequence of operations and the enrollment functions are identical to those given in sections 3.4.2, 3.4.3 and 3.4.4. The only difference lies in the actual recognition step: As shown in Table 26, the verification call accepts an explicit claim of identity.

Table 26 – Verification against an enrolled identity

Prototype	int32_t verify_template(	
	const uint8_t *verification_template,	Input
	const uint32_t verification_template_size,	Input
	const uint32_t enrolled_identity_claim,	Input
	double *similarity);	Output
Description	This function searches a template against the enrollment set, and outputs a list of candidates. The returned similarity is a non-negative distance measure. When either the input template or the enrolled data for the claimed identity are the result of the failed template generation, the similarity score shall be -1 and the function return value shall be 2.	
Input Parameters	identification_template	A template from convert_multiface_to_identification_template().
	identification_template_size	The size, in bytes, of the input verification template $0 \leq N \leq 2^{16} - 1$
	enrolled_identity_claim	An integer index into the enrollment set. The face represented by the verification template is claimed to be that of this enrolled identity. The value of this parameter is an index into the array of enrolled identities passed into the finalize_enrollment function of section 3.4.4.
Output Parameters	similarity	A similarity score resulting from comparison of the templates, on the range [0,DBL_MAX]. See section 2.3.5.
Return Value	0	Success
	2	Either or both of the input templates were result of failed feature extraction
	Other	Vendor-defined failure

3.6. Pose conformance estimation

3.6.1. Overview

The functions of this section support testing of whether a face in an image has frontal pose. This supports conformance testing of, for example, the Full Frontal specification of the ISO standard [ISO]. The goal is to support a marketplace of products for acquisition time assessment of pose. This is important because pose is arguably the most influential covariate on face recognition error rates, and is not generally controllable by design of the acquisition system.

NIST encourages participants in this study to implement real-time video rate implementations, and also slower more accurate methods. The test and API is not intended to cover multi-frame techniques (e.g. from video) nor tracking.

The functional specification here supports a DET analysis in which false-rejection of actually frontal images can be traded off against false acceptance of non-frontal images via a frontal-conformance parameter, t. The exact meaning of the "frontality" value returned by this function is not regulated by the NIST specification. However a reasonable implementation would embed a monotonic relationship between the output value and non-frontal angle (i.e. compound rotation involving azimuthal head yaw and pitch).

The formal ISO requirement is for five degree rotation in pitch and yaw. While the ISO standard establishes an eight degree limit on roll angle, this is of less importance. NIST will not consider roll angle.

3.6.2. API

Table 27 provides a function for computing a pose conformance measurement from an image. Although the function makes use of the MULTIFACE structure (for consistency with the rest of the API), the function will ordinarily be invoked with just a single image.

Table 27 - Pose conformance estimation

Prototypes	int32_t estimate_frontal_pose_conformance(	
	const MULTIFACE *input_faces,	Input
	double *non_frontalities);,	Output
Description	This function takes a MULTIFACE, and outputs a non-frontality value for each image. The images of the MULTIFACE will be independent - i.e. they will generally not be frames from a video. The non-frontality value should increase with larger deviations from frontal pose.	
Input Parameters	input_faces	An instance of a Table 10 structure.
Output Parameters	non-frontalities	For the i-th input image, the i-th output value will indicate how from frontal the head pose is.
Return Value	0	Success
	2	Elective refusal to process this kind of MULTIFACE
	4	Involuntary failure to extract features (e.g. could not find face in the input-image)
	6	Elective refusal to produce a template (e.g. insufficient pixels between the eyes)
	8	Cannot parse input data (i.e. assertion that input record is non-conformant)
	Other	Vendor-defined failure. Failure codes must be documented and communicated to NIST with the submission of the implementation under test.

3.7. Software and Documentation

3.7.1. SDK Library and Platform Requirements

Participants shall provide NIST with binary code only (i.e. no source code). Header files (".h") are allowed, but these shall not contain intellectual property of the company nor any material that is otherwise proprietary. It is preferred that the SDK be submitted in the form of a single static library file (ie. ".lib" for Windows or ".a" for Linux). However, dynamic and shared library files are permitted.

The core library shall be named according to Table 28. Additional dynamic or shared library files may be submitted that support this "core" library file (i.e. the "core" library file may have dependencies implemented in these other libraries).

Table 28 - Implementation library filename convention

Form	libMBE_provider_class_sequence.ending				
Underscore delimited parts of the filename	libMBE	provider	class	sequence	ending
Description	First part of the name, required to be this.	Single word name of the main provider EXAMPLE: Acme	The single letter class identifier give in Table 3. EXAMPLE: C	A two digit identifier to be incremented every time a SDK is emailed to NIST. EXAMPLE: 07	One of .so .a .dll .lib
Example	libMBE_Acme_C_07.a				

NIST will report the size of the supplied libraries.

3.8. Configuration and vendor-defined data

The implementation under test may be supplied with configuration files and supporting data files. The total size of the SDK, that is all libraries, include files, data files and initialization files shall be less than 50MB.

Editor's NOTE: Is this sufficient?

3.8.1. Linking

NIST will link the provided library file(s) to various ISO 98/99 "C/C++" language test driver applications developed by NIST. Participants are required to provide their library in a format that is linkable using "gcc" with the NIST test driver, which is compiled with gcc version 4.1. These use libc. The link command might be:

```
        gcc -I. -Wall -m64 -o mbetest  mbetest.c  -L.  -lMBE_Acme_C_07 -ljpeg -lpng -lpthread
```

NIST has also successfully used "g++" for linking, but note that the API calls itself must have "C" linkage. The prototypes of this document will be written to a file "mbe.h" which will be included via

```
extern "C"
{
#include <mbe.h>
}
```

NIST will link to JPEG and PNG libraries, sourced, respectively from http://www.ijg.org/ and see http://libpng.org.

All compilation and testing will be performed on x86 platforms. Thus, participants are strongly advised to verify library-level compatibility with gcc (on an equivalent platform) prior to submitting their software to NIST to avoid linkage problems later on (e.g. symbol name and calling convention mismatches, incorrect binary file formats, etc.).

Dependencies on external dynamic/shared libraries such as compiler-specific development environment libraries are discouraged. If absolutely necessary, external libraries must be provided to NIST upon prior approval by the Test Liaison.

3.8.2. Installation and Usage

The SDK must install easily (i.e. one installation step with no participant interaction required) to be tested, and shall be executable on any number of machines without requiring additional machine-specific license control procedures or activation.

The SDK’s usage shall be unlimited. The SDK shall neither implement nor enforce any usage controls or limits based on licenses, execution date/time, number of executions, presence of temporary files, etc.

The SDK must be installable using simple file copy methods. It must not require the use of a separate installation program.

3.8.3. Documentation

Participants shall provide complete documentation of the SDK and detail any additional functionality or behavior beyond that specified here. The documentation must define all (non-zero) vendor-defined error or warning return codes.

3.8.4. Modes of operation

Individual SDKs provided shall not include multiple “modes” of operation, or algorithm variations. No switches or options will be tolerated within one library. For example, the use of two different “coders” by an feature extractor must be split across two separate SDK libraries, and two separate submissions.

3.8.5. Watermarking of images

The SDK functions shall not watermark or otherwise steganographically mark up the images.

3.9. Runtime behavior

3.9.1. Interactive behavior

The SDK will be tested in non-interactive “batch” mode (i.e. without terminal support). Thus, the submitted library shall not use any interactive functions such as graphical user interface (GUI) calls, or any other calls which require terminal interaction e.g. reads from “standard input”.

3.9.2. Error codes and status messages

The SDK will be tested in non-interactive “batch” mod, without terminal support. Thus, the submitted library shall run quietly, i.e. it should not write messages to "standard error" and shall not write to “standard output”.

3.9.3. Exception Handling

The application should include error/exception handling so that in the case of a fatal error, the return code is still provided to the calling application.

3.9.4. External communication

Processes running on NIST hosts shall not side-effect the runtime environment in any manner, except for memory allocation and release. Implementations shall not write any data to external resource (e.g. server, file, connection, or other process), nor read from such. If detected, NIST will take appropriate steps, including but not limited to, cessation of evaluation of all implementations from the supplier, notification to the provider, and documentation of the activity in published reports.

3.9.5. Stateful behavior

All components in this test shall be stateless. This applies to segmentation, feature extraction and matching. Thus, all functions should give identical output, for a given input, independent of the runtime history. NIST will institute appropriate tests to detect stateful behavior. If detected, NIST will take appropriate steps, including but not limited to, cessation of evaluation of all implementations from the supplier, notification to the provider, and documentation of the activity in published reports.

4. References

FRVT 2002	Face Recognition Vendor Test 2002: Evaluation Report, NIST Interagency Report 6965, P. Jonathon Phillips, Patrick Grother, Ross J. Micheals, Duane M. Blackburn, Elham Tabassi, Mike Bone
FRVT 2002b	Face Recognition Vendor Test 2002: Supplemental Report, NIST Interagency Report 7083, Patrick Grother
AN27	NIST Special Publication 500-271: American National Standard for Information Systems — *Data Format for the Interchange of Fingerprint, Facial, & Other Biometric Information – Part 1*. (ANSI/NIST ITL 1-2007). Approved April 20, 2007.
MINEX	P. Grother et al., *Performance and Interoperability of the INCITS 378 Template*, NIST IR 7296 http://fingerprint.nist.gov/minex04/minex_report.pdf
MOC	P. Grother and W. Salamon*, MINEX II - An Assessment of ISO/IEC 7816 Card-Based Match-on-Card Capabilities* http://fingerprint.nist.gov/minex/minexII/NIST_MOC_ISO_CC_interop_test_plan_1102.pdf
PERFSTD	ISO/IEC 19795-4 — Biometric Performance Testing and Reporting — Part 4: Interoperability Performance Testing. Posted as document 37N2370. The standard was published in 2007, and can be purchased from ANSI at http://webstore.ansi.org/ or ISO.
ISO	ISO/IEC 19794-5:2005 — Information technology — Biometric data interchange formats — Part 5: Face image data. The standard was published in 2007, and can be purchased from ANSI at http://webstore.ansi.org/ or ISO
STD05	ISO/IEC 19794-5:2005 — *Information technology — Biometric data interchange formats — Part 6: Face image data* The standard was published in 2005, and can be purchased from ANSI at http://webstore.ansi.org/ or ISO.
INTEROP	ISO/IEC 19795-4 — Biometric Performance Testing and Reporting — Part 4: Interoperability Performance Testing.

Annex A
Application to participate in MBE-STILL

A.1 Who should participate

Providers of face recognition technologies are invited to participate in MBE. In addition, companies, research organizations, or universities that have developed mature prototypes or who research face recognition are invited to participate.

The algorithms and software need not be "operational," nor a production system, nor commercially available. However, the system must, at a minimum, be a stable implementation capable of being "wrapped" (formatted) in the API specification that NIST has specified in section 1.17 for this evaluation.

Anonymous participation will not be permitted. This means that signatories to this Agreement acknowledge that they understand that the results (see sections 1.14 and Annex A.7) of the evaluation of the software and/or hardware will be published with attribution to their organization(s).

A.2 How to participate

Those wishing to participate in MBE testing must do all of the following, on the schedule listed on Page 2.

— Indicate via email a non-binding "Intention to Participate" - see the schedule on Page 2.

— Request an SDK ID from NIST (for use per section 3.1).

— Follow the instructions for cryptographic protection of your SDK here. http://face.nist.gov/mbe/crypto_protection.pdf

— Send a signed and fully completed copy of this entire Annex A, including the *MBE Application to Participate* form (TO BE RELEASED LATE 2009). This must identify, and include signatures from, the Responsible Parties as defined in section A.4.

— Provide an SDK (Software Development Kit) library which complies with the API (Application Programmer Interface) specified in this document.

The *MBE Application to Participate* shall be sent to:

MBE Test Liaison National Institute of Standards and Technology Information Access Division (894) 100 Bureau Drive A203/Tech225/Stop 8940 Gaithersburg, MD 20899-8940 USA	In cases where a courier needs a phone number please use NIST shipping and handling on: 301 -- 975 -- 6296.

A.3 NIST activity

A.3.1 Initiation

Upon completion of the application procedure, the organization shall be classified as a "Participant".

A.3.2 Supplier validation

Registered Participants will be provided with a small Validation Dataset available on the website http://face.nist.gov/mbe. Prior to submission of their SDK, the Participant must to verify that their software executes on the validation data, and produces correct similarity scores and templates.

A.3.3 Submission of software to NIST

NIST requires that all software submitted by the participants be signed and encrypted. Signing is done with the participant's private key, and encrypting is done with the NIST public key, which is published on the MBE Web site. NIST will validate all submitted materials using the participant's public key, and the authenticity of that key will be verified using the key fingerprint. This fingerprint must be submitted to NIST by writing it on the signed participation agreement.

By encrypting the submissions, we ensure privacy; by signing the submission, we ensure authenticity (the software actually belongs to the submitter). **NIST will not accept into MBE any submission that is not signed and encrypted. NIST accepts no responsibility for anything that is transmitted to NIST that is not signed and encrypted with the NIST public key.**

The detailed commands for signing and encrypting are given here: http://face.nist.gov/mbe/crypto_protection.pdf

A.3.4 Acceptance testing

Software submitted shall implement the MBE API Specification of section 3.

Upon receipt of the SDK and validation output, NIST will attempt to reproduce the same output by executing the SDK on the validation imagery, using a NIST computer. In the event of disagreement in the output, or other difficulties, the Participant will be notified.

A.3.5 Limits of testing

NIST will use the Participant's SDK software only for purposes related to the testing described in this document. The provided software will also be used to resolve any errors identified subsequent to the test or publication of results. NIST agrees not to use the Participants software for purposes other than indicated herein, without express permission by the Participant. NIST reserves the right to conduct analyses of the output data and measurements beyond those described in this document. NIST reserves the right to apply the software to images from sensors not enumerated in this document.

A.3.6 Third part analysis

Outputs of the test runs (e.g. similarity scores, candidate lists) may be supplied to U. S. Government organizations who sponsor the test. Such data may in turn be provided to third party organizations. NIST will not associate such data with the names of the SDK provider.

A.4 Parties

A.4.1 Responsible Party

The Responsible Party is an individual with the authority to commit the organization to the terms in this document.

A.4.2 Point of contact

The Point of Contact is an individual with detailed knowledge of the system applying for participation.

The MBE Liaison is the government point of contact for MBE. All correspondence should be directed to mbe2010@nist.gov, which will be received by the MBE Liaison and other MBE personnel.

These correspondences may be posted on the FAQ (Frequently Asked Questions) area of the http://face.nist.gov/mbe at the discretion of the MBE Liaison. The identity of those persons or organizations whose correspondences lead to FAQ postings will not be made public in the FAQ.

A.5 Access to MBE validation data

The MBE Validation Data is supplied to Participants to assist in preparing for MBE.

The images in the MBE Validation Data are representative of the MBE Test Data only in their format. Image quality, collection device and other characteristics are likely to vary between the Validation and Test Datasets.

A.6 Access to MBE test data

The MBE Test Datasets are in some cases protected under the Privacy Act (5 U.S.C. 552a), and will be treated as Sensitive but Unclassified and/or Law Enforcement Sensitive.

MBE Participants shall have no access to MBE Test Data, either before, during or after the test.

A.7 Reporting of results

A.7.1 Reports

The Government will combine appropriate results into one or more MBE reports. Together these will contain, at a minimum, descriptive information concerning MBE, descriptions of each experiment, and aggregate test results. NIST will include

— DET performance metrics as the primary indicators of one-to-one verification accuracy,

— ISO/IEC 19795-4 interoperability matrices as the primary measures of interoperability, and

— Image generation, template generation, and matching timing statistics.

NIST may compute and report other aggregate statistics. NIST intends to publish results in one or more NIST Interagency Reports. The reports will contain

— contain the names of participants,

— contain the results of all participants' implementations with attribution to the participants.

A.7.2 Pre-publication review

Participants will have an opportunity to review and comment on the reports. Participants' comments will be either incorporated into the main body of the report (if it is decided NIST reported in error) or published as an addendum. Comments will be attributed to the participant.

A.7.3 Citation of the report

Subsequent to publication of our reports Participants may decide to use the results for their own purposes. Such results shall be accompanied by the following phrase: "Results shown from the Multiple Biometric Evaluation (MBE) do not constitute endorsement of any particular system by the U. S. Government." Such results shall also be accompanied by the URL of the MBE Report on the MBE website, http://face.nist.gov/mbe.

A.7.4 Rights and ownership of the data

Any data generated, deduced, measured or otherwise obtained during MBE (excepting the submitted SDK itself), as well as any documentation required by the Government from the participants, becomes the property of the Government. Participants will not possess a proprietary interest in the data and/or submitted documentation.

A.8 Return of the supplied materials

NIST will not return any supplied software, documentation, or other material to vendors.

A.9 Agreement to participate

COMPLETE DETAILS OF HOW TO FORMALLY PARTICIPATE IN THE MBE WILL BE ADDED TO THIS SECTION LATE 2009.